Book of Kali:
Foundational Wireless Attacks

Author: Seccult
Date: June 21’st 2026
Version: V1.0

Table of Contents:

Introduction: - pages 1-2

Attacking Wireless Networks Overview: - pages 3-7

Attacking WEP Wireless Networks Overview: - pages 8

Theory Behind WEP Cracking + Notes: - pages 8-9

Passive Client Based WEP Cracking Using “aircrack-ng” Tutorial: - pages 9-11

WEP Fake Authentication Tutorial: - pages 11-14

WEP Cracking Using an ARP Request Replay Attack Tutorial: - page 14-16

WEP Cracking Using a Korek Chopchop Attack Tutorial: - pages 16-20

WEP Cracking Using a Fragmentation Attack Tutorial: - page 20-23

Fragmentation Versus Korek Chopchop Attacks : - page 24

WEP SKA Authentication Attack Tutorial: pages 25-32

Aircrack and Crunch WPA2 Tutorial: pages 32-38

Practical “aircrack-ng” and “crunch” Command Examples: page 38-39

“aircrack-ng” and “rockyou.txt” Wordlist Tutorial: page 39-40

“crunch” Command Options: page 40-42

Cracking WPS Pins with “reaver” Tutorial: 42-46

Using “reaver” or “bully” to Perform a WPS Pixiedust Attack Tutorial: 46-49

Accessing an A.P. Upon Successful WPS Attack with “reaver”, or “bully”: Page 49

Exploiting Router Vulnerabilities via “exploit-db.com”: pages 50-51

Exploting Router Vulnerabilities via “nmap”: pages 51-54

Best Router Security Practices and Suggestions: page 55

Best Network Security Practices and Suggestions: pages 55-58

Conclusion: page 58-59

Introduction:

whoami: So here is the deal, I am a multi certified pentester, and defense analyst, for years I've survived as a scrappy freelancer in the industry.

I very recently took a job offer that spoke to my heart, but prior to this position I turned down several six figure job offers because they sounded fucking retarded, and I think I woulda kissed the end of a shotgun had I taken them.

I'm a high school dropout, never went to no fancy university, my set of knowledge came from being self taught, and weaponizing my autism.

When I started my journey, it came out of a place of love for privacy and freedom, I wanted to understand how these systems worked, and that was enough to carry me through.
These guides/tutorials have been painstakingly converted from my original handwritten notes, and as I complete converting more I will upload them here free of charge.

I'm doing this because when I was starting out I had a difficult time getting help, and I got put down so much, some IRL people I knew actively discouraged me from even attempting to learn. Now I am in a position where I'm living my life doing the thing I love, for reasons I feel good about, and I reached something people spend their entire lives seeking, fulfilment.

If I listened to everyone telling me I couldn't, or shouldn't do something, I wouldn't be here, I said fuck em instead, and I did the thing a lot of people thought I couldn't do, it wasn't always easy, but I did it. I'm writing all this out to give some context, and to say you can learn about these systems from zero, you don't need to go to university, or pay exorbitant amounts to do so, and if you genuinely want to learn, these are my notes free of charge to help you.

Book of Kali Wireless Attacks Overview:

This third manual is meant for people starting their journey from zero practical linux experience, it's not a theortical course, and builds directly off of the knowledge from my first manual **The Book Of Kali: Basics**, and the followup **The Book Of Kali: Privacy Fundamentals**

That said, this manual puts an emphasis on how to conduct foundational WIFI audits against the most common wireless security protocols using Kali Linux, and helps one familiarize themselves with some of the built in tools, commands, and pentesting frameworks that can be used for these purposes.

Beyond these subjects, it also covers some mitigation suggestions that can be used to harden a Wifi network, and many of these concepts can be applied towards other penetration testing disciplines in the future, including nmap reconnaissance, exploit research (and usage), as well as password cracking methodologies.

Unlike my previous manual which was meant specifically for beginners, this manual may also be of interest for those who already hold linux fundamentals, and understand how to operate a terminal,

but want some insight on how “wireless attacks” are conducted. This document will also serve as a reference manual for how these attacks may be performed, and will be of particular use for those seeking to challenge the **OSWP exam by Offsec**.

Note: Offsec no longer provides a .pdf file, or videos for the newest iteration of the OSWP, and while some of the newer attacks are not covered in this document, I will include them later in a followup to this course which discusses Advanced Wi-Fi attacks.

This may be my pre-ultimate free course, but I plan to continue publishing free cyber security content on medium, and I hope you will continue with me on my journey, thank you for reading.

Prerequisites:

The lessons, and tutorials in this course require one to own **packet injecting wireless cards**, as well as a **vulnerable router** for the WEP, and WPA attacks (some of these attacks will require **two wireless cards** to be utilized in tandem). One will need to do research on which router to buy, but the following are recommendations for wireless cards one can use, these are ones I can vouch for:

- **Panda PAU07 N600 Wireless N USB Adapter**

Link: https://www.amazon.com/dp/B00U2SIS0O/?tag=pandaw-20

- “Plug in and play” NIC resource list:

https://github.com/morrownr/USB-WiFi/blob/main/home/The_Short_List.md

Note: For those who can't afford wireless cards, or a vulnerable router right away, there will be some free wifi labs posted in the resources section.

Lastly, this tutorial was written with Kali in mind, you can use another OS, and it should be okay, but you may have issues following certain sections if you do. Unless one is running the Kali OS bare metal, one will need to configure their system so it works with thier external wireless card.

One more thing:

Please feel free to offer some suggestions, but just be aware I'm releasing these guides for free, and I may not be able to make changes right away (or at all), sorry, I’m fucking crazy busy.

Note about copywrite: This document is free to read, download, share, just link to the official source, and please give me credit.

Attacking Wireless Networks Overview:

In the following chapters, one will learn the fundamentals of how to attack wireless networks, the theory behind how these attacks work, as well as a bit of post exploitation, before finishing off with mitigation suggestions to help secure a wifi network.

This chapter will primarily cover the different vulnerabilities of each of the common network protocols. This chapter will also give one a brief description of these network protocols, and an overview of the main tools used to attack them.

Overview of network protocols:

WEP: This stands for **Wired Equivalent Privacy** and is the original Wi-Fi security standard, this is the easiest kind of network protocol to compromise.

Note: The **WEP** protocol has been phased out, as such finding **WEP** connections will be very difficult. In instances where these protocols are found they are likely a result of 'legacy rot", a term used to describes out of date systems still being used as a cost saving measure, or because updating these systems will cause issues with other legacy systems.

WPA: This stands for **Wif-Fi Protected Access**, and was the Wi-Fi Alliances direct response, and replacement to the vulnerabilities of the WEP standard, WPA was adopted as the standard in 2003.

Most attacks against WPA are conducted on a supplementary system that was rolled out with WPA, being **Wi-Fi Protected Setup (WPS)**, which was designed to make it easy to link devices to modern access points, **PSK (Pre-Shared Key) vulnerabilities** also exist against WPA implementations.

WPA2: As of 2006, WPA has been officially superseded by WPA2, which stands for **Wi-Fi Protected Access II**.

One of the most significant changes between WPA, and WPA2 is the mandatory use of **AES algorithms**, and the introduction of **CCMP (counter cipher mode with block chaining message authentication code protocol)**, as a replacement for **TKIP**. However **TKIP** is still preserved in WPA2 as a fallback system, and for interoperability with WPA.

WPA2's biggest vulnerability remains a **(WPS)** pin attack, however **PSK (Pre-Shared Key) vulnerabilities** also exist against WPA2 implementations.

WPA3: This is the newest wireless security protocol, and drastically increases the security of a network, some vulnerabilites still exist as it is possible to perform a **dragonblood** attack against some of the implementations of the WPA3 protocol, and downgrade the protocol so one may attack it like a **WPA2** implementation.

The other aspect of WPA3 is that due to the costs of the routers that support this protocol, it is not common to see implemented on most enviorments.

Overview of the "aircrack-ng suite", and other Wi-fi hacking tools:

The following is a brief overview of several wi-fi hacking tools, and what these tools may be used for, **airgeddon** will get its own chapter later, this list should not be considered comprehensive.

airmon-ng: This tool is needed in nearly every Wi-Fi attack, and is used to change ones network card between both **promiscuous**, and **monitor** modes, meaning one will be able to see, and receive all network traffic.

airodump-ng: This tool enables one to scan, and capture packets of their specification, and is particularly useful in password cracking.

aircrack-ng: This is the primary application of the **aircrack-ng suite** of tools, it's used for password cracking. It's capable of using statistical techniques to crack WEP, and dictionary, and bruteforce attacks against WPA, and WPA2 implementations after capturing the "WPA handshake".

aireplay-ng: This tool can be used to generate, or accelerate traffic on the **AP (Access Point)**. This can be very useful in **deauthentication attacks**, to bump every user of an AP. As well as for use in WEP, and WPA2 **password attacks**, **ARP injection**, and **replay attacks**.

airdecap-ng: Enables one to decrypt wireless traffic once they have cracked the key. In other words, once one has the key of a wireless access point, not only can they use the bandwidth on the access point, but utilizing **airdecap-ng** they can **decrypt** all the traffic on the AP, and observe everything that's being done on the network (the key is used for both access, and encyption).

airtun-ng: This tool is a **virtual tunnel interface creator**. One may use **airtun-ng** to set up an **IDS (Intrusion Detection System)** on the wireless traffic to detect malicious, and other kinds of traffic on the wireless access point. For instance, if one were looking to receive an alert for a particular type of traffic, they could use **airtun-ng** to set up a virtual tunnel that connects to an **IDS** like **snort** to send one alerts.

airolib-ng: This tool stores, and manages **ESSID's (the name of the access point)**, and password lists that will assist in speeding up the WPA/WPA2 password cracking process.

airbase-ng: This tool enables one to turn their laptop, and wireless card into an AP. This can be especially useful when performing a **rouge access point attack**, or **evil twin attack**.

Basically, **airbase-ng** allows one to attack the clients (rather than the AP), and encourages the clients to associate with the imitation AP one set up, rather than the real AP.

Note: There are other **aircrack-ng suite** tools, however these are the primary ones used for Wi-Fi pentesting.

Fun Fact: The **ng** at the end of the **aircrack-ng suite** name, and at the end of each associated tool stands for "new generation" as **aircrack-ng** replaced the original **aircrack** tools found on older **backtrack linux** OS's.

Other Essential Tools to Attack Wireless Networks:

crunch: This tool is used to generate wordlists, where one can specify a standard character set, or a character set one specifies, this tool is able to generate all possible combinations, and permutations.

wash: This is a scanning tool, that informs one which wireless interfaces are vulnerable to a **pin attack** using **reaver**, along with the wireless connection strength.

reaver: This tool implements a **brute force attack** against **Wi-Fi Protected Setup (WPS)** registrar PINS in order to recover WPA/WPA2 passphrases, as described in this document:

https://sviehb.wordpress.com/wp-content/uploads/2011/12/viehboeck_wps.pdf

The **reaver** tool has been designed to provide robust, and practical auditing attacks against WPS, and has been tested against a wide variety of access points, and WPS implementations.

On average **reaver** will recover the target AP's plain text WPA/WPA2 passphrase in 4-10 hours, depending on the AP. In practice, it will generally require half this time to guess the correct WPS pin, and recover the passphrase, there is a total of **11000 possible WPS pins combinations**.

bully: This tool works much in the same way as the **reaver** tool against WPS implementations, however it makes several advancements that make the tool more efficient, effective, and can be considered the successor to the **reaver** tool.

airgeddon: This is a wifi penetration testing framework, somewhat akin to **metasploit**, in contains a variety of attacks that can be carried out **automatically** (instead of manually). Another unique aspect of the framework, is that it carries some unique attacks not found in the **aircrack-ng suite** of tools, such as some social engineering vectors, and wps pin database attacks.

Alternative WPA2 password cracking tools: There are also a plethora of other Wi-fi tools that can be discussed such as **pyrate**, **john**, and **coWPatty**, to crack passwords. Most of these tools perform similarly to **aircrack-ng**, however in some instances may be more useful.

Explaination between "monitor", and "promiscuous" modes:

Monitor Mode: Allows a computer with a wireless network interface controller (WNIC) to monitor all traffic received from the wireless network. Monitor mode allows packets to be captured without having to associate with an **access point**, or **ad hoc network** first. Monitor mode **only** applies to wireless networks.

Promiscous Mode: This mode is similar to **monitor mode** and is also used for packet sniffing, however unlike **monitor mode**, in **promiscuous mode** one can do so on both wired and wireless networks. Promiscous mode unlike monitor mode can only accept packages meant for it, where an association has already been made with an **access point**, or **ad hoc network**.

Note: "Monitor mode", and "Promiscuous mode" are two of eight modes that 802.11 wireless cards

can operate in, (or technically nine).

Scan Field Names:

The following is a list of the scan field names, and a description of what they mean, and are used for, these are important to understand when conducting network reconnaissance, and preparing to perform Wi-Fi attacks.

BSSID: This is the MAC address of that specific router/access point.

PWR: This is the strength of the signal, the lower the number the better the connection is.

Beacons: Number of announcement packets sent by the A.P. Each access point sends about 10 beacons per second at the lowest rate (1M), so they can usually be picked up from very far.

 # Data: Number of captured data packets (if WEP unique IV count), including data broadcast packets.

#/s: Number of data packets per second, measured over the last 10 seconds.

CH: Channel number (taken from beacon packets). Sometimes packets from other channels are captured even if **airodump-ng** is not hopping because of radio interference.

MB: Maximum speed supported by the A.P. if MB=11, it's 802.11b, if MB=22 it's 802.11b+, and higher rates are 802.11g. The dot (after 54 above) indicates short preamble is supported. Will display **e** following MB speed value if the network has QoS enabled.

ENC: Encryption algorithm in use:

- **OPN**: no encryption being used.
- **WEP?:** WEP or higher (not enough data to choose between WEP, and WPA/WPA2).
- **WEP:** Without the question mark indicates static, or dynamic WEP.
- **WPA, or WPA2:** If TKIP, or CCMP is present.

CIPHER: This will dispmay the cipher detected, which will consist of the following:

- **CCMP**
- **WRAP**
- **TKIP**
- **WEP**
- **WEP40**
- **WEP104**

Not mandatory, but TKIP is typically used with WPA, and CCMP is typically used with WPA2. WEP40 is displayed when the key index is greater than 0. The standard states that the index can be 0-3 for 40bit, and should be 0 for 104 bit.

AUTH: This field represents the authentication protocol used.

- **MGT**
- **WPA/WPA2** (using a seperate authentication server).
- **SKA** (shared key for **WEP**)
- **PSK** (pre-shared key for **WPA/WPA2)**
- **OPN** (open for **WEP**)

ESSID: This field displays the wireless network name, the so called "**SSID**", which can be empty is **SSID** hiding is activated.

WPS Scan Field Names:

The following is a list of the WPS scan field names, and a description of what they mean, and are used for, these are important to understand when conducting WPS network reconnaissance, and preparing to perform WPS based Wi-Fi attacks.

BSSID: This is the MAC address of that specific router/access point.

CH: Channel number (taken from beacon packets). Sometimes packets from other channels are captured even if **wash** is not hopping because of radio interference.

dbm: This is the unit of levels used to indicate that a power ratio is expressed in decibles (**db**) with reference to one **milliwatt**.

This is important to keep in mind, the lower the **dbm** number reads, the better ones connection to an A.P. will be; and thus the easier it will be to perform a bruteforce WPS pin attack with **reaver**.

For instance, a **dbm** of -20, is significantly better than a **dbm** of -20.

WPS: This is the version of **Wi-Fi Protected Setup** the A.P. has. The lower the version, the more susceptible, and vulnerable the A.P. will be to an attack.

Lck: This field displays whether a A.P. WPS is locked.

Note: In order to use **reaver** to attack the access point, this field **must read "No"**. Otherwise a WPS attack with **reaver** will be impossible! The **Lck** field is the most important field to examine before attempting a **reaver** WPS attack!

Vendor: The vendor field displays the make of the router. This information is useful if one intends to attempt a WPS **Pixiedust** attack.

ESSID: Shows the wireless network name. The so called "**SSID**", which can be **empty** if **SSID hiding** is activated.

Attacking WEP Wireless Networks Overview:

The following chapters, will be covering the different methods used to attack **WEP networks**. While it should be noted that WEP (or Wired Equivalent Privacy) is a depreciated Wi-Fi "standard", the different vulnerabilites are still important to learn in the event one needs to compromise a WEP network in "the wild".

Note: Wep has been replaced by WPA, WPA2, and the current newest standard for Wi-Fi security WPA3.

WEP Attack Methods:

1). Passive Client based WEP cracking using **aircrack-ng**.

2). ARP Request Replay Attack. *

3). Korek ChopChop Attack. *

4). Fragmentatiom Attack. *

* **Note:** All these attacks require both a "Fake Authentication", as well as the use of either the **aircrack-ng suite** of tools, or the **airgeddon** framework.

Theory Behind WEP Cracking + Notes:

The **WEP** security protocol uses the **RC4 cipher**, this cipher generates pseudorandom bytes to **XOR (mix)** with ones data to create the encypted data. It generates these using an algorithm that "shuffles" the numbers **0-255** around in an array.

The initial arrangement of this array is derived from the key, by what is referred to as a "Key Scheduling Algorithm". In certain cases, this key scheduling algorithm can be worked backwards from the output of the **RC4 (the pseudo-random data)** if one can figure out enough about what the state of the array is, they will be able obtain the key.

It is not secure to use the same key twice when conducting symmetric crypto, so an initialization **vector (just extra data that is different for each message)** is included. This **IV (initialization vector)** is not a secret, and is usually sent along with the message in **plaintext**, since one would have to know the rest of the key anyway.

However in **RC4**, certain **IV's** cause working the "Key Scheduling Algorithm" backwards to be much easier.

The use of **fake-auth** attacks, **deauth**, ect, that can be executed with the **aircrack-ng suite** of tools, are performed to force machines on the network to generate more packets; because that increases the

chance a packet will have a “weak” IV, that in turn can be leveraged to compromise the Wi-Fi security and obtain the **plaintext password**.

The reason a lot of weak IV’s are required is so one can perform the same attack multiple times, at which point the correct byte value will appear much more often than any other value, and the password can then be deduced.

Notes on WEP:

- According to statistics provided by **wigle.net**, about **2.56%** of Wi-Fi networks still use WEP for encyption as of April 17’th, 2026.
source: **https://wigle.net/stats**

- WEP was depreciated in 2004.

- There are 2 different kinds of WEP encryption, the standard **64 bit**, and the slightly more secure **128 bit** encyption.

Passive Client Based WEP Cracking Using “aircrack-ng” Tutorial:

Overview: This chapter will discuss the most basic (and time consuming) method of cracking a WEP key.

This is achieved by **passive packet capturing** on a “client based” WEP network; once enough time has passed (and enough packets captured) one will be able to crack the WEP key!

To do so one may use the following steps:

Step One | Turn off the network card using **ifconfig**:

To set ones card into **monitor mode**, and proceed with the rest of this tutorial, one must first turn off their network card with the **ifconfig** command.

To do so, one can use the following command format example:

ifconfig [network interface] down

Step Two | Setting ones network card to “monitor mode”:

Now that one has disabled their wireless network card, they may now set up their card in **monitor mode** by using the **iwconfig** command.

To do so, one can use the following command format example:

iwconfig [network interface] mode monitor

Step Three | Turn the network card back on using **ifconfig**:

Once one has set their network card to **monitor mode**, they must then turn it back on.

To do so, one can use the following command format example:

ifconfig [network interface] up

Step Four | Using **airmon-ng** to kill processes:

Once ones network card is back up in **monitor mode**, one should now begin to kill processes in order to ensure they will dont interfere with the attack later, one must first check for these processes, then kill them individually.

To view potentially conflicting processes one may use the following command format:

airmon-ng check [network interface]

Followed by killing these processes, via the following command format:

kill [process name]

Once one has enemerated the potentially conflicting processes, and killed them, they may then proceed to the next step.

Step Five | Perform a network scan using **airodump-ng**:

Once one has killed all the potentially interfering processes, they may now use **airodump-ng** to conduct a network scan.

To do so, one can use the following command format example:

airodump-ng [network interface]

Once the scan is running, one should inspect the scan output for a target network utilizing **WEP encryption**. Upon discovering a suitable network to attack, one should take note of the following:

- The network BSSID
- The network Channel Number

As this information will be required to perform the next step in the attack.

Step Six | Using **airodump-ng** to capture WEP network IV's, and create a .cap file:

Once one has found a suitable WEP network to target, one can then capture the target network IV's

and create a .cap file for **aircrack-ng** to utilize later to crack the key.

Note: One can only capture IV's this way if at least one client is connected to the WEP network!

To do so, one can use the following command format example:

airodump --bssid [target bssid #] -c [target channel #] -w [name of the target .cap file] [network interface]

Practical command format example:

airodump-ng --bssid 00:22:6B:71:60:FF -c 6 -w defcon wlan0

Step Seven | Using **aircrack-ng** to crack the WEP password from the target .cap file:

Once one has captured the target WEP .cap file, they may now leverage the **aircrack-ng** tool against the file to attempt to crack the network key.

To do so, one can use the following command format example:

aircrack-ng [target WEP .cap file path]

Note: At a minimum one needs at least **6000 IV's** to crack basic WEP encryption, if one does not have sufficient IV's **aircrack-ng** will suggest a new number of IV's to try again later (once one has captured enough).

WEP Fake Authentication Tutorial:

Overview: The previous chapter disscussed how one can crack a WEP key on a client based network using **aircrack-ng**.

However the previous method will not work if no clients are connected to the target network; in order for one to attack a WEP network without any clients present, one must first perform a **fake authentication attack** with **aireplay-ng**.

A **fake authentication** must be performed first prior to any clientless based attack method taking place, that is what will be covered in this chapter.

Step One | Turn off the network card using **ifconfig**:

To set ones card into **monitor mode**, and proceed with the rest of this tutorial, one must first turn off their network card with the **ifconfig** command.

To do so, one can use the following command format example:

ifconfig [network interface] down

Step Two | Setting ones network card to “monitor mode”:

Now that one has disabled their wireless network card, they may now set up their card in **monitor mode** by using the **iwconfig** command.

To do so, one can use the following command format example:

iwconfig [network interface] mode monitor

Step Three | Turn the network card back on using **ifconfig**:

Once one has set their network card to **monitor mode**, they must then turn it back on.

To do so, one can use the following command format example:

ifconfig [network interface] up

Step Four | Using **airmon-ng** to kill processes:

Once ones network card is back up in **monitor mode**, one should now begin to kill processes in order to ensure they will dont interfere with the attack later, one must first check for these processes, then kill them individually.

To view potentially conflicting processes one may use the following command format:

airmon-ng check [network interface]

Followed by killing these processes, via the following command format:

kill [process name]

Once one has enemerated the potentially conflicting processes, and killed them, they may then proceed to the next step.

Step Five | Perform a network scan using **airodump-ng**:

Once one has killed all the potentially interfering processes, they may now use **airodump-ng** to conduct a network scan.

To do so, one can use the following command format example:

airodump-ng [network interface]

Once the scan is running, one should inspect the scan output for a target network utilizing **WEP**

encryption. Upon discovering a suitable network to attack, one should take note of the following:

- The network BSSID
- The network Channel Number

As this information will be required to perform the next step in the attack.

Step Six | Perform a "fake authentication" using **aireplay-ng**:

Once one has performed a network scan of the WEP network they want to attack, and have taken note of its **BSSID**, one can now use **aireplay-ng** to perform a **fake authentication attack**.

To do so, one can use the following command format example:

aireplay-ng --fakeauth 0 -a [target BSSID] - h [local MAC address] [network interface]

Practical command format example:

aireplay-ng --fakeauth 0 -a E0:69:95:B8:BF:77 -h 00:c0:ca:6c:ca:12 wlan0

The following is an explanation of the above command format:

aireplay-ng This command specifize that this tool will be used for this attack.

--fakeauth This command option is used to specify the type of attack that will be performed against the WEP network (in this instance a **fake authentication attack**. The **0** that proceeds this command option specifize the number of packets that will be sent.

-a [target BSSID] Is where one would input the target BSSID obtained during the prior network scan.

-h [local MAC address] Is where one would input their MAC address, so they may authenticate with the target network.

Note: To find out what ones MAC address is, one can use the following command format:

ifconfig [network interface]

Upon running the above command, one can locate their MAC address next to the **HWaddr** field.

[network interface] This field is where one would input the name of the local network interface being used, for example **Wlan0**.

Upon executing the **fake authentication attack**, **aireplay-ng** will let you know if the attack was successful.

WEP fake authentication notes:

- When running a **airodump-ng** scan against a specific WEP network without clients; underneath the **AUTH** field of the **airodump-ng** scan it will be blank.

Once one has connected via a **fake authentication attack** the space under the **AUTH** field should read **OPN**.

WEP Cracking Using an ARP Request Replay Attack Tutorial:

Overview: To utilize the **ARP Request Replay** attack method after successfully associating with the target AP, one will need to wait for an ARP packet, one will then capture this packet, and "inject" it into the target network traffic; this will force the AP to generate a new ARP packet containg a new IV.

One will then capture this new packet, and inject it into the traffic again; this process is repeated until the number of IV's captured is sufficient enough to crack the key.

ARP Request Replay Attack Prerequisites:

Before an ARP Request Replay Attack can be attempted, one must first perform a **fake authentication** against the target WEP network.

One can refer to the prior chapter titled **WEP fake authentication tutorial**, for detailed instructions on how a "fake authentication" is performed.

Once one has successfully performed a "fake authentication", they may then proceed with this tutorial.

ARP Request Replay Attack Tutorial:

The following steps can be used to perform this attack once one has successfully performed a **fake authentication** with the target AP:

Step One | Utilizing **airodump-ng** to create a .cap file of the target WEP network:

Once one has performed a **fake authentication** with the desired target WEP network; they may then run a detailed network scan, and also create a **.cap** file for later use with **aircrack-ng** to crack the WEP key.

Command format example:

airodump-ng --bssid [target BSSID #] -c [target channel #] -w [name of the target .cap file] [network interface]

Note: For a more detailed explanation of this step please refer to **Step Six | Using airodump-ng to capture WEP network IV's**, in the **Passive Client Based WEP Cracking Using "aircrack-ng" Tutorial**

chapter.

Step Two | Run an ARP Request Replay Attack using **aireplay-ng**:

Note: Please keep the terminal with the detailed WEP scan running.

To run an ARP Request Replay attack against the target network one must start a **new terminal session** (there should be two terminals running in tandem), in this new terminal they may then use the following command.

ARP Request Replay Attack Command Format Example:

aireplay-ng --arpreplay -b [target BSSID #] -h [local MAC address] [network interface]

Practical command format example:

aireplay-ng --arpreplay -b E0:69:95:B8:BF:77 -h 00:c0:ca:12 wlan0

The following is an explaination of the above command:

aireplay-ng This command specifize that this tool will be used for this attack.

--arpreplay This command specifize the kind of attack method that will be used, in this case an **aireplay attack**.

-b [target BSSID #] Is where one would input the target BSSID obtained during the prior network scan.

-h [local MAC address] Is where one would input their MAC address, so they may authenticate with the target network.

Note: To find out what ones MAC address is, one can use the following command format:

ifconfig [network interface]

Upon running the above command, one can locate their MAC address next to the **HWaddr** field.

[network interface] This field is where one would input the name of the local network interface being used, for example **Wlan0**.

Once one runs the ARP Request Replay attack, they should observe the terminal session containing the **airodump-ng** scan of the target network; if the attack was successful, the number of IV's being transmitted should increase substantially, these IV's will in turn be captured.

Step Three | Using **aircrack-ng** to crack the WEP password from the .cap file:

Once one has successfully performed the ARP Request Replay attack, they may now use the **aircrack-**

ng tool to attempt to crack the WEP key from their previously captured .cap file from the target network.

Command format example:

aircrack-ng [target WEP .cap file path]

Note: For a more detailed explanation of this step please refer to **Step Seven | Using aircrack-ng to crack the WEP password from the .cap file**, in the **Passive Client Based WEP Cracking Using "aircrack-ng" Tutorial** chapter.

WEP Cracking Using a Korek Chopchop Attack Tutorial:

Overview: Using the **Korek Chopchop attack** method, one will capture an ARP packet, and then attempt to guess its key stream; and use it to forge a new packet (using **packetforge-ng**). Afterwards one can inject this new forged packet into the target traffic to generate new IV's.

Korek Chopchop Attack Prerequisites:

Before an Korek Chopchop attack can be attempted, one must first perform a **fake authentication** against the target WEP network.

One can refer to the prior chapter titled **WEP fake authentication tutorial**, for detailed instructions on how a "fake authentication" is performed.

Once one has successfully performed a "fake authentication", they may then proceed with this tutorial.

Korek Chopchop Attack Tutorial:

The following steps can be used to perform this attack once one has successfully performed a **fake authentication** with the target AP:

Step One | Utilizing **airodump-ng** to create a .cap file of the target WEP network:

Once one has performed a **fake authentication** with the desired target WEP network; they may then run a detailed network scan, and also create a **.cap** file for later use with **aircrack-ng** to crack the WEP key.

Command format example:

airodump-ng --bssid [target BSSID #] -c [target channel #] -w [name of the target .cap file] [network interface]

Note: For a more detailed explanation of this step please refer to **Step Six | Using airodump-ng to**

capture WEP network IV's, in the **Passive Client Based WEP Cracking Using "aircrack-ng" Tutorial** chapter.

Step Two | Run a Chopchop attack to capture a keystream file using **aireplay-ng**:

Note: Please keep the terminal with the detailed WEP scan running.

One may now run a Chopchop attack against the target network to capture, and create a keystream file.

To do so, one may use the following command format:

aireplay-ng --chopchop -b [target BSSID #] -h [local MAC address] [network interface]

Practical command format example:

aireplay-ng --chopchop -b E0:69:95:B8:BF:77 -h 00:c0:ca:6c:ca:12 wlan0

The following is an explaination of the above command:

aireplay-ng This command specifize that this tool will be used for this attack.

--chopchop This command specifize the kind of attack method that will be used, in this case a **Korek Chopchop attack**.

-b [target BSSID #] Is where one would input the target BSSID obtained during the prior network scan.

-h [local MAC address] Is where one would input their MAC address, so they may authenticate with the target network.

Note: To find out what ones MAC address is, one can use the following command format:

ifconfig [network interface]

Upon running the above command, one can locate their MAC address next to the **HWaddr** field.

[network interface] This field is where one would input the name of the local network interface being used, for example **Wlan0**.

Once one runs the **Korek Chopchop attack**, and it captures an ARP packet; one will then be given the option to decrypt the ARP packets keystream, when prompted with the question "**Use this packet?**"

One should select **y (for yes)** and press the **enter** key.

Afterwards the attack will resume, and attempt to determine the keystream for the target access point.

Once the attack completes running it will save the keystream as an **.xor file**, one should take note of this as they will require it later; this is used in order to create a **forged packet** using **packetforge-ng**.

Note: Once the attack completes running it may not completely determine the keystream, this is okay, it is still possible to make this attack work with an incomplete keystream; however an incomplete keystream is not guaranteed to succeed.
Step Three | Forging a fake packet using **packetforge-ng**:

To forge a fake ARP packet for later use for **aireplay-ng**, one must use the **.xor** file created in the previous step.

To do so, use the following command format:

packetforge-ng -0 -a [target BSSID #] -h [local MAC address] -k 255.255.255.255 -l 255.255.255.255 -y [input .xor file created from last step] -w [name of the forged packet being created]

Practical command format example:

packetforge-ng -0 -a E0:69:95:B8:BF:77 -h 00:c0:ca:6c:ca:12 -k 255.255.255.255 -l 255.255.255.255 -y replay_dec-0824-110731.xor -w wepforgedpacketname

The following is an explanation of the above command.

packetforge-ng This command specifize that this tool will be used for the purpose of forging a packet.

-0 Specifies that one will be forging an ARP packet.

-a [target BSSID #] Is where one would input the target A.P. MAC address.

-h [local MAC address] This command, and input field is where on would input their local MAC address.

-k This command is used to set the destination IP.

Note: Most access points do not care what IP is used for the destination and/or source IP, it is common to use **255.255.255.255**

-l This command is used to set the source IP.

Note: Most access points do not care what IP is used for the destination and/or source IP, it is common to use **255.255.255.255**

-y [input .xor file created from last step] This command, and input field is where one would input the file path to the.xor file created from the **chopchop attack** in the last step.

-w [name of the forged packet being created] This command, and input field is where one would write the desired name of the forged packet file that is to be created.

Step Four | Injecting the forged packet into the target traffic:

In this step one will be injecting the forged packet they created in the previous step into the target traffic.
By doing so one will cause the data to increase substantially, allowing one to then capture IV's quicker; and therefore crack the WEP password faster.

To do so, use the following command format:

aireplay-ng -2 -r [forged packet file from previous step] [network interface]

Practical command format example:

aireplay -2 -r WEPforgedpacketname wlan0

The following command when executed will inject the forged packet into the traffic; however before doing this one will be given a final prompt asking if this is what they would like to do.

When asked "use this packet?" enter **y** for yes, and press the **enter** key to run the command.

The following is an explanation of the above command:

aireplay-ng This command specifize that this tool will be used for this attack.

-2 This command is used to delineate a specific packet for replaying (injecting).

-r [forged packet file from the previous step] This command and input field specify the file that will be used to inject into traffic.

[network interface] This field is where one would input the name of the local network interface being used, for example Wlan0.

Step Five | Using **aircrack-ng** to crack the WEP password from the .cap file:

Once one has successfully performed the **korek chopchop attack**, they may now use **aircrack-ng** to attempt to crack the WEP key.

One can attempt to crack the WEP key from their previously captured **.cap** file of the target network.

Command format example:

aircrack-ng [target WEP .cap file path]

Note: For a more detailed explanation of this step please refer to **Step Seven | Using aircrack-ng to crack the WEP password from the .cap file**, in the **Passive Client Based WEP Cracking Using "aircrack-ng" Tutorial** chapter.

WEP Cracking Using a Fragmentation Attack Tutorial:

Overview: Using the fragmentation attack method, one will need to obtain **1500 bytes** of the **PGRA (Pseudo Random Generation Algorithm)**, this in turn can be used to forge a new packet; which can then be injected into the target traffic to generate new IV's.

While similar to a **Korek Chopchop attack** in both command format, and function, a **Fragmentation attack** does differ in some key ways:

- A Fragmentation attack will work extremely fast when successful.

- On the downside one is required to be physically closer to the access point, since if any packets are lost the attack will fail.

Note: More information on the differences between a **Korek Chopchop Attack**, and a **Fragmentation Attack,** can be found in the chapter titled **Fragmentation versus Chopchop Attacks**.

Fragmentation Attack Prerequisites:

Before a Fragmentation Attack can be attempted one must first perform a **fake authentication** with the target WEP network.

Please read the prior chapter titled **WEP fake authentication tutorial**, for detailed instructions on how a **fake authentication** is performed.

Once one has performed a **fake authentication** they may then proceed with this tutorial.

Step One | Utilizing **airodump-ng** to create a .cap file of the target WEP network:

Once one has performed a **fake authentication** with the desired target WEP network; they may then run a detailed network scan, and also create a **.cap** file for later use with **aircrack-ng** to crack the WEP key.

Command format example:

airodump-ng --bssid [target BSSID #] -c [target channel #] -w [name of the target .cap file] [network interface]

Note: For a more detailed explanation of this step please refer to **Step Six | Using airodump-ng to capture WEP network IV's**, in the **Passive Client Based WEP Cracking Using "aircrack-ng" Tutorial** chapter.

Step Two | Run a Fragmentation attack to capture a PRGA packet using **aireplay-ng**:

Note: Please keep the terminal with the detailed WEP network scan running.

To run a **Fragmentation attack** against the target network to capture a packet in order to determine that packets PRGA one can use the following.

Command Format Example:

aireplay-ng --fragment -b [target BSSID #] -h [local MAC address] [network interface]

Practical Command Format Example:

aireplay-ng --fragment -b E0:69:95:B8:BF:77 -h 00:c0:ca:6c:ca:12 mon0

The following is an explanation of the above command:

aireplay-ng This command specifize that this tool will be used for this attack.

--fragment This command specifize the kind of attack method that will be used, in this case a **Fragmentation attack**.

-b [target BSSID #] Is where one would input the target BSSID obtained during the prior network scan.

-h [local MAC address] Is where one would input their MAC address, so they may authenticate with the target network.

Note: To find out what ones MAC address is, one can use the following command format:

ifconfig [network interface]

Upon running the above command, one can locate their MAC address next to the **HWaddr** field.

[network interface] This field is where one would input the name of the local network interface being used, for example **Wlan0**.

Once one runs the **Fragmentation attack** and it captures a packet, they will then be given the option to try to determine the packets PRGA. When prompted with the question **use the packet?**, type in **y** for **yes**.

Note: One may need to try several times before they obtain a packet that the **Fragmentation attack** can extrat the PRGA from; one should be patient, and continue to repeat attempts until they succeed.

Once one succeeds they will be informed that the keystream has been saved in a **.xor file**, take note of the file name, as one will require it to forge a packet in the next step with the **packetforge-ng** tool.

Step Three | Forging a fake ARP packet using **packetforge-ng**:

To forge a fake ARP packet for later use with **aireplay-ng**, one must use the **.xor** file created in the previous step.

To do so, use the following command format:

packetforge-ng -0 -a [target BSSID #] -h [local MAC address] - 255.255.255.255 -l 255.255.255.255 -y [input .xor file created from last step] -w [name of the forged packet being created]

Practical Command Format Example:

packetforge-ng -0 -a E0:69:95:B8:BF:77 -h 00:c0:ca:6c:ca:12 -k 255.255.255.255 -l 255.255.255.255 -y fragment-0824-111939.xor -w fragmentforgedpacketname

The following is an explanation of the above command:

packetforge-ng This command specifize that this tool will be used for the purpose of forging a packet.

-0 Specifies that one will be forging an ARP packet.

-a [target BSSID #] Is where one would input the target A.P. MAC address.

-h [local MAC address] This command, and input field is where on would input their local MAC address.

-k This command is used to set the destination IP.

Note: Most access points do not care what IP is used for the destination and/or source IP, it is common to use **255.255.255.255**

-l This command is used to set the source IP.

Note: Most access points do not care what IP is used for the destination and/or source IP, it is common to use **255.255.255.255**

-y [input .xor file created from last step] This command, and input field is where one would input the file path to the.xor file created from the **fragmentation attack** in the last step.

-w [name of the forged packet being created] This command, and input field is where one would write the desired name of the forged packet file that is to be created.

Step Four | Injecting the forged packet into the target traffic:

In this step one will be injecting the forged packet they created in the previous step into the target

traffic.

By doing so one will cause the data to increase substantially, allowing one to then capture IV's quicker; and therefore crack the WEP password faster.

To do so, use the following command format:

aireplay-ng -2 -r [forged packet file from previous step] [network interface]

Practical command format example:

aireplay -2 -r WEPforgedpacketname wlan0

The following command when executed will inject the forged packet into the traffic; however before doing this one will be given a final prompt asking if this is what they would like to do.

When asked "use this packet?" enter **y** for yes, and press the **enter** key to run the command.

The following is an explanation of the above command:

aireplay-ng This command specifize that this tool will be used for this attack.

-2 This command is used to delineate a specific packet for replaying (injecting).

-r [forged packet file from the previous step] This command and input field specify the file that will be used to inject into traffic.

[network interface] This field is where one would input the name of the local network interface being used, for example **Wlan0**.

Step Five | Using **aircrack-ng** to crack the WEP password from the .cap file:

Once one has successfully performed the **Fragmentation attack**, one can now use **aircrack-ng** to attempt to crack the WEP key.

One can attempt to crack the WEP key from their previously captured .cap file of the target network.

Command Format Example:

aircrack-ng [target WEP .cap file path]

Note: For a more detailed explanation of this step please refer to **Step Seven | Using aircrack-ng to crack the WEP password from the .cap file**, in the **Passive Client Based WEP Cracking Using "aircrack-ng" Tutorial** chapter.

Fragmentation Versus Korek Chopchop Attacks:

Overview: There are differences between **Fragmentation**, and **Korek Chopchop attacks**, this chapter discusses some of the pros, and cons of each, so one may make an informed decision as to what attack is best suited to a particular use case.

Fragmentation Attack Pros:

- Can obtain the full packet length of 1500 bytes XOR. This means one can subsequently create any size of packet.

- May work where a Chopchop attack does not.

- Is extremely fast, it yields the XOR stream rapidly when successful.

Fragmentation Attack Cons:

- Setup to execute the attack is more subject to the device drivers. For instance, **Atheros** does not generate the correct packets unless the wireless card is set to the MAC address one is spoofing.

- One needs to be physically closer to the access point since if any of the packets are lost, then the attack will fail.

Chopchop Attack Pros:

- One can be physically farther from the access point and still successfully perform the attack.

- May work where a Framentation attack does not.

Chopchoo Attack Cons:

- Cannot be used against every access point.

- The maximum XOR bits is limited to the length of the packet one performs the Chopchop attack against.

- Much slower than the Fragmentation attack.

WEP SKA Authentication Attack Tutorial:

Overview: Within the context of the **Wireless Equivalent Privacy** (**WEP**) protocol, **Shared Key Authentication** (**SKA**) is a legacy verification method that could be used in conjunction with **WEP** to add an additional layer of security to the protocol; as its purpose is to verify the identify of a device attempting to connect to a network.

The theory of the SKA process as provided by Th4ntis Cybersec is listed below:

1). The client sends an authentication request to the Access Point (AP)
2). The AP replies with a clear-text challenge.
3). The client encrypts the challenge-text using the configured WEP key and sends it back in another authentication request.
4). The AP decrypts the response. If this matches the challenge text, the AP sends back a positive reply

Further Reading: The following blog gives more information on wireless protocols, and thier associated security features https://cybersec.th4ntis.com/wireless/wireless-basics

In theory this should make **WEP** protocols that leverage **SKA**, in practice however, it is relatively trivial for an attacker to bypass this security implementation using a mixture of techniques previously discussed in the last chapters.

This section provides a tutorial on how one can defeat **SKA** authentication security, and can be considered a capstone challenge against **WEP** protocol WiFi hacking.

Attacking WEP SKA "aircrack-ng":

Note: This tutorial does not discuss command specifics at length, because it assumes the operator has already familiarized themselves with how these commands function from the prior chapters on **WEP** attacks.

Step One | Turn off the network card using **ifconfig**:

To set ones card into **monitor mode**, and proceed with the rest of this tutorial, one must first turn off their network card with the **ifconfig** command.

To do so, one can use the following command format example:

ifconfig [network interface] down

Practical Command Format Example:

ifconfig wlan0 down

Step Two | Setting ones network card to "monitor mode":

Now that one has disabled their wireless network card, they may now set up their card in **monitor mode** by using the **iwconfig** command.

To do so, one can use the following command format example:

iwconfig [network interface] mode monitor

Practical Command Format Example:

iwconfig wlan0 mode monitor

Step Three | Turn the network card back on using **ifconfig**:

Once one has set their network card to **monitor mode**, they must then turn it back on.

To do so, one can use the following command format example:

ifconfig [network interface] up

Practical Command Format Example:

ifconfig wlan0 up

Step Four | Using **airmon-ng** to kill processes:

Once ones network card is back up in **monitor mode**, one should now begin to kill processes in order to ensure they will dont interfere with the attack later, one must first check for these processes, then kill them individually.

To view potentially conflicting processes one may use the following command format:

airmon-ng check [network interface]

Practical Command Format Example:

airmon-ng check wlan0

Followed by killing these processes, via the following command format:

kill [process name]

Once one has enemerated the potentially conflicting processes, and killed them, they may then proceed to the next step.

Step Five | Perform a network scan using **airodump-ng**:

Once one has killed all the potentially interfering processes, they may now use **airodump-ng** to conduct a network scan.

To do so, one can use the following command format example:

airodump-ng [network interface]

Practical Command Format Example:

airodump-ng wlan0

Once the scan is running, one should inspect the scan output for a target network utilizing **WEP encryption**. Upon discovering a suitable network to attack, one should take note of the following:

- The network BSSID
- The network Channel Number
- The target ESSID
- The target Protocol
- Whether any data is being transmitted through the target network (useful to easily determine whether there are any connected clients).

As this information will be required to perform the next step in the attack against a **WEP** protocol.

Step Six | Perform a target specific network scan using **airodump-ng**:

Once a "broad" scan has been conducted, one may now leveage the information from this scan to perform a more detailed targeted scan, in this case against a "WEP SKA" network.

Command Format Example:

airodump-ng --bssid [target bssid #] -c [target channel #] -w [desired name for the capture file] [network interface device name]

Practical Command Format Example:

airodump-ng --bssid 00:14:D1:E1:C7:62 -c 11 -w target wlan0

Upon running this detailed scan one should take note as to any whether there are any **connected clients** via the MAC addresses under the "**station header**".

Important note: The terminal that executed this scan should **not** be closed, it needs to be kept running to determine if the attack is working, and so a **target.cap** file is created, and can be used to crack the password later.

Step Seven | Attempt a "**Fake Authentication**" attack using **aireplay-ng**:

As one should be able to determine earlier from the scans we conducted against the target network, if the target network is using **WEP encryption**, the first course of action is to attempt a "**Fake Authentication**" attack using the **aireplay-ng** tool.

Note: Prior to running the following commands, one should open a **new** terminal session prior to doing so.

Command Format Example:

aireplay-ng --fakeauth 0 -a [target bssid #] -h [local network interface MAC address] [network interface name]

Practical Command Format Example:

aireplay-ng --fakeauth 0 -a 00:14:D1:E1:C7:62 -h 00:0d:a0:cd:87:ce wlan0

Note: The **--fakeauth 0** command option is used to specify a **fake authentication** attack will take place **indefinitely**.

If successful one will receive a "**successful authentication message**", and the blank **AUTH header** in the terminal window where one was performing the prior **airodump-ng** scan of the target WEP network, should change to "**OPN**", or "**SKA**".

Important Note: For the purpose of this tutorial, in this example the target AUTH header changed to "**SKA**" once the attack was executed, which denied the operator access to authenticate. As such, in the next step one will attempt to bypass that form of authentication.

Step Eight | Deauthenticating clients to capture a **PRGA.xor** file to bypass SKA, using **aireplay-ng**:

Due to the operator being denied access in the last step due to the SKA, in this step one will need to perform a **deauthentication attack** using **aireplay-ng** against a connected client to capture a PRGA.xor file to bypass the SKA in the following step.

This can be achieved by using the information discovered from the previous target **airodump-ng** scan, to find a client "MAC address" to deauthenticate.

Note: Use the following commands in the same terminal session the previous **Fake Authentication** attack was conducted from.

Command Format Example:

aireplay-ng -0 0 -a [target bssid #] -c [target MAC address] [network interface name]

Practical Command Format Example:

aireplay-ng -0 0 -a 00:14:D1:E1:C7:62 -c 00:02:72:8E:19:5E wlan0

Note: The **-0 0** command specifies a deauthentication attack will be run, and the following **0** specifies it will be run indefinitely.

Once one runs the attack, if successful, the client will be disconnected from thier session, and one will be **informed that a keystream file has been captured** on the terminal window running the target **airodump-ng** scan.

Using the captured **.xor file** we can now bypass the SKA, and fake authenticate in the next step.

Note: For the sake of operational security one should stop the attack once the .xor file has been captured by pressing **CTRL+Z** to stop the deauthentication attack. Allowing the attack to run indefinitely could cause an attacker to be discovered in a real world pentesting scenerio.

Step Nine | Bypassing the SKA, using the captured .xor file, and a "Fake Authentication" attack using **aireplay- ng**:

Once one has captured the .xor file, one can confirm the .xor file name for use with **aireplay-ng**, by using the **ls** command in the same directory location as the terminal session where the capture attack took place.

The name of the .xor file captured in this example is "**target-01-00-14-D1-E1-C7-62.xor**" .

Using this .xor file one can now bypass the SKA using a revised "**Fake Authentication**" attack with **aireplay-ng**.

Note: Use the following commands in the same terminal session the previous "**Deauthentication"** attack was conducted from.

Command Format Example:

aireplay-ng --fakeauth 0 -a [target bssid #] -h [local network interface MAC address] -y [.xor file used to bypass the SKA] [network interface name]

Practical Command Format Example:

aireplay-ng --fakeauth 0 -a 00:14:D1:E1:C7:62 -h 00:0d:a0:cd:87:ce -y target-01-00-14-D1-E1-C7-62.xor wlan0

Once one runs the attack, if successful, one should now be able to connect to the target via "**Fake Authentication**", if successful an "**Association successful :-)**" message will appear.

Important Note: One may eventually be disconnected, if this occurs one can simply reconnect via performing another "**Fake Authentication**".

Step Ten | Attacking the network using a "Korek Chop Chop" attack to capture a .xor file, using **aireplay-ng**:

Once successfully authenticated, one can now proceed to attack the WEP network using the **aireplay-ng** tool, in order to force the creation of packets one can use to capture IV's which one can later utilize to crack the WEP key.

In this case one would be able to use a "**Korek CHOP CHOP**" attack to achieve this using the **aireplay-ng** tool.

Note: Use the following commands in a **new terminal session**, while keeping the previous terminsl sessions active.

Command Format Example:

aireplay-ng --chopchop -b [target bssid #] -h [local network interface MAC address] [network interface name]

Practical Command Format Example:

aireplay-ng --chopchop -b 00:14:D1:E1:C7:62 -h 00:0D:A0:CD:87:CE wlan0

Note: The **--chopchop** command is used to specify a **Korek Chop Chop** attack is to be used.

Once the attack has been executed, packets will be reviewed, and when a potentially suitable .xor file is presented one will be given the option to capture it by typing in "**y**" when prompted, this is the file that will be used later with **packetforge-ng** to **forge a ARP packet**, which one can use later to inject into the network to force the creation of IV's.

Note: In this case because one has already captured an .xor file earlier, that specific file will used in the next step.

Step Eleven | Using the captured .xor file, to forge a ARP package using **packetforge-ng:**

Once one has a suitable .xor file, one can now attempt to create a forged ARP file using **packetforge- ng.**

Command Format Example:

packetforge-ng -0 -a [target bssid #] -h [local network interface MAC address] -k [input the destination IP here] -l [input the source IP here] -y [input .xor file captured from last step] -w [desired name of the forged packet being created]

Practical Command Format Example:

packetforge-ng -0 -a 00:14:D1:E1:C7:62 -h 00:0d:a0:cd:87:ce -k 255.255.255.255 -l 255.255.255.255 -

y target-01-00-14-D1-E1-C7-62.xor -w fakearp

Note: The **-0** command specifies that a ARP packet is to be forged.

Once created, one can now use the forged packet in the next step, for the puposes of this tutorial the forged package should be titled **fakearp**.

Step Twelve | Injecting our forged packet using **aireplay-ng**:

Now one can inject the previously forged ARP packet, using **aireplay-ng** to force the creation of IV's for later use to crack the WEP key.

Note: The following command should be used from the same terminal session where the previous "**Fake Authentication"** took place.

Command Format Example:

aireplay-ng -2 -r [the name of the forged ARP packet to be injected] [network interface name]

Practical Command Format Example:

aireplay-ng -2 -r fakearp wlan0

Note: The **-2** command specifies an "**injecting**" attack

The following command when executed will inject the forged packet into the traffic; however before doing this one will be given a final prompt asking them if this is what they would like to do.

When asked "**use this packet?**" enter "**y**" for yes, and press the "**enter**" key to run the command. Note: Once to attack has been executed, one can check the previous target **airodump-ng** scan, to see whether the data packets are increasing rapidly, if so the attack is a success, and the next step can be taken.

Step Thirteen | Cracking the WEP key using **aircrack-ng**:

Once one has determined that the packets are increasing rapidly, one can now use the previous **target .cap file** created in the **airodump-ng** scan of the target to crack the WEP key.

That specific file was saved to the directory where the scan took place, one can determine the exact file name of the **.cap** file created by using the **ls** command is the said scan directory location.

Note: For the purposes of this tutorial the target .cap file created is called "**stage2-01.cap**".

So now that one knows the file name, they can proceed to use the **aircrack-ng** tool to crack the target WEP.

Command Format Example:

aircrack-ng -0 [target .cap file]

Practical Command Format Example:

aircrack-ng -0 target-01.cap

Note: The **-0** command option is a special colorful mode for **aircrack-ng**.
If the above command works, then the key will be decrypted using a statistical bruteforce attack, and **aircrack-ng** will inform the operator as such.

Aircrack and Crunch WPA2 Tutorial:

Overview: This tutorial will demonstrate how to perform a **brute force password attack** against a wireless network using **aircrack-ng**, and the **crunch** tools. Some of the same initial steps needed to perform this attack, will be used again later when performing a **reaver** PIN bruteforce attack.

Step One | Finding out ones local wireless network card name using the **ifconfig** command:

First one needs to input the following into their terminal session:

ifconfig

This command will return the status of the currently active network interfaces, using this command one can determine what their wireless interface is (in this case the wireless network interface nane is **wlan0**).

Note: Depending on what OS one is using the wireless network interface name may be different, so one should determine what theirs is and adapt accordingly.

Step Two | Turn off the network card using **ifconfig**:

To proceed with the rest of the tutorial one must turn off their wireless network card.

Command Format Example:

ifconfig [network interface] down

Practical Command Format Example:

ifconfig wlan0 down

The above command will turn off the **wlan0** wireless network interface card, so one may be able to modify it afterwards.

Step Three | Setting the network card to monitor mode:

Once one has turned off their wireless network card, they may now set it to **monitor mode** by typing **iwconfig** followed by the wireless network interface name, which would then be followed by the **mode** command, and lastly the **monitor** command option.

Command Format Example:

iwconfig [network interface] mode monitor

Practical Command Format Example:

iwconfig wlan0 mode monitor

The above command would set the **wlan0** network interface to **monitor mode**.

Step Four | Turn the network card back on using **ifconfig**:

Once one has set their wireless interface card to **monitor mode**, they can then turn it back on by typing **ifconfig** into the terminal, followed by the wireless network interface name, and lastly the **up** command option.

Command Format Example:

ifconfig [network interface] up

Practical Command Format Example:

ifconfig wlan0 up

The above command will turn on the **wlan0** interface.

Step Five | Using **airmon-ng** to kill processes:

Once ones network interface card is back up in **monitor mode**, one may then type the following command into the terminal **airmon-ng check**, followed by their wireless network interface name.

Command Format Example:

airmon-ng check [network interface]

Practical Command Format Example:

airmon-ng check wlan0

Using the previous command will now bring up a list of processes associated with the **wlan0** network

interface card that could cause problems.

The next step is to kill these processes, this should be done in a specific order, otherwise the processes may be able to respawn one another.

To kill a process utilizing **airmon-ng**, one needs to type in **kill** followed by the **PID number** of the process they would like to kill. For instance **kill 1048** would kill the process associated with the 1048 number.

Command Format Example:

kill [PID #]

Practical Command Format Example:

kill 1048

Note: Once a process has been killed one needs to restart the **airmon-ng check** process to kill the rest.

Order to kill processes:

1). NetworkManager
2). dhclient

The rest of the processes can be killed in any order.

Step Six | Perform a network scan using **airodump-ng**:

Once one has killed all the processes (and done a final check using **airmon-ng** to ensure they are all dead), they may now use **airmon-ng** to perform a network scan.

To do so one may type in the **airodump-ng** command into the terminal, followed by the wireless interface name.

Command Format Example:

airodump-ng [network interface]

Practical Command Format Example:

airodump-ng wlan0

Using the above command will now provide one with a scan of all the networks within proximity.

Step Seven | Perform a specific scan using **airodump-ng** to achieve a four way handshake:

Once one has performed a general scan, and has found a specific network they wish to attack, they may do so using the following.

Command Format Example:

airodump-ng -c [channel #] --bssid [BSSID #] -w [scan file name] [network interface]

Practical Command Format Example:

airodump-ng -c 1 --bssid 00:22:6B:71:60:FF -w mosesscan wlan0

This is a breakdown of the above command.

airodump-ng This is the tool used to perform the scan.

-c [channel #] This command, and following field is used specify the specific channel of the network one wishes to attack.

--bssid [BSSID #] This command, and the following field is used to specify the BSSID (aka MAC address) of the network that is being specified for the attack.

-w [scan file name] This command, and the following input field is used to write the scan capture information to a file whose name you specify (after the **-w** command).

[network interface] This is the name of the network interface one previously set to **monitor mode** (for instance **wlan0**).

In order for a **four way handshake** to occur, a disconnected device must connect to the network while one is using the **airodump-ng** command, this will generally take a substantial amount of time. However the process can be sped up by performing a successful deauthentication attack.

Note: The scan capture information can be found in the **/root** directory, and these files will be named after whatever one chose to call them after the **-w** command. These file names will have the following extensions:

- .cap
- .csv
- .kismet.csv
- .kismet.netxml

Step Eight | Using **aireplay-ng** to perform a wi-fi deauthentication attack, to force a “four way handshake”:

Instead of waiting for a device to connect to the network to acquire a **four way handshake**, it is possible for one to force this connection by performing a deauthentication attack.

Note: In order to perform this attack one must at least have one device listed under the **STATION** field.

Prerequisite: It is very important to have seperate terminal sessions open, one terminal session so one may perform the **deauth** attack with **aireplay-ng**, and anither seperate terminal session so one can watch for the handshake using the previous **airodump-ng** command outlined in the previous step.

To perform the attack, one would need to type the following command format into the terminal:

aireplay-ng -0 0 -a [target BSSID] [network interface]

Practical Command Format Example:

aireplay-ng -0 0 -a 00:22:6B:71:60:FF wlan0

The following is an explanation of the above command:

aireplay-ng This command specifize that this tool will be used for this attack.

-0 This command option specifize that deauthentication packets are to be sent to one or more clients which are currently associated with a particular access point.

0 This command option is used to specifize to send deauthentication requests continuously in an infinite loop (alternatively one may specify a specific number of deauth packets to send).

-a [BSSID] This command, and the following field is used to specify the BSSID (aka MAC address) of the network that is being specified for the deauthentication attack.

[network interface] This is the name of the network interface one previously set to monitor mode (for instance **wlan0**).

Note: The **-c** command option may also be used to perform a deauthentication attack against a specific mac address of a client, if this is omitted then all clients will be deauthenticated.

Once one has run the attack, they should wait for a minute or two to make sure all devices have been deauthenticated.

This is important, as one will not recieve any output that the devices have been deauthenticated, and as such one will need to allow the attack to run for a moment to ensure that the devices are in fact being deauthenticated.

Once one is satisfied, they may stop the attack, bu using the following key combination:

Ctrl+Z

This will then allow one to capture a **four-way handshake**, once the deauthenticated devices reconnect to the network.

Note: If one does not cancel the attack, it will run continuously, and no devices will be able to connect to the network, this will cause suspicion if allowed to run indefinitely.

Step Nine | Using **aircrack-ng** in combination with **crunch** to perform a bruteforce password attack:

It is very important to note that the following attack will only be effective if one has some idea of what the password of the network they are attacking may be.

For instance:

- How many characters the password may contain.
- Where the characters in the password may be.
- What specific characters are being used.

The more information one has, the more likely the password can be defeated.

Note: Without some idea of what the password parameters are, a bruteforce attack will be near impossible to perform, and could literally take years to crack the password, even utilizing good hardware.

That said, the following information in this step will explain some of the different options available to help one perform this attack, a breakdown of the attack command, what it means, and how it functions.

Lastly some command examples will also be included.

The following is an example of a crunch, and aircrack-ng command format example that can be used to perform a bruteforce attack:

crunch [min password length] [max password length] -t [known password characters, or specified character options] | aircrack-ng -w - [capture file] -e [essid, or network name]

The following is an explanation of the above command:

crunch This command specifize that this tool will be used for this attack.

[min password length] This field is where one would input a number to represent the minimum amount of characters they feel the password contains (for example, if one feels the password only contains five characters, they would input the number **5**).

[max password length] This field is where one would input a number to represent the maximum amount of characters they feel the password contains (for example, if one feels the password contains a maximum of eight characters, they would input the number **8**).

-t This command is used to specify a pattern within the password, e.g. **@@god@@@@** where only the (**@ , s %** and **^**) symbols will change.

| This is a pipe, which is used to pass the input from **crunch** to another program (in this case **aircrack-ng**).

aireplay-ng This command specifies that this tool will be used to help crack the password.

-w This command option specifies that a wordlist attack is to be used.

- [capture file] The following command **stdin** (aka **standard input**), and the following input field is used to specify the **capture file** of the network one wishes to attack (this is the **.cap** file of the network that one has obtained the **four-way handshake** from.

-e This command option specifies that an **ESSID** will be used to target the network.

[essid, or network name] This input field is used to specify the name of the network that is going to be attacked.

Practical “aircrack-ng” and “crunch” Command Examples:

Overview: This chapter provides several command format examples of how the **crunch** tool can be used in tandem with **aircrack-ng** to crack Wi-Fi passwords.

Practical Command Format (Example One):

crunch 8 8 -t %%%%%007 1234567890 | aircrack-ng -w - CanCom_1686-01.cap -l CanCom_1686_key -e CanCom_1686

crunch 8 8 This command tells **crunch** to run a minimum, and maximum password length of **8 characters**.

-t This command option then specifies that the **%** symbols will insert numbers into the password pattern from the following character set (**1234567890**).

|This pipe symbol passes the crunch output to the **aircrack-ng** tool.

aircrack-ng -w - This command specifies that **standard input** will be used with the tool.

CanCom_1686-01.cap is the specified capture file containing the **four-way handshake**, of the target network one is going to attack.

-l CanCom_1686_key This command will write the network key once found to a file called **CanCom_1686_key**.

-e CanCom_1686 This command specifies the name of the network being attacked.

“aircrack-ng” and “rockyou.txt” Wordlist Tutorial:

Overview: Instead of using **crunch** with **aircrack-ng**, to perform a bruteforce password attack, it is possible to use the **aircrack-ng** tool in conjunction with a wordlist file to perform a **dictionary** attack.

rockyou.txt Wordlist Background, and Usage Tutorial:

The **rockyou.txt** file is often touted by hackers as the “gold stardard” wordlist used when conducting dictionary based attacks. The wordlist contains over **14341564 unique plaintext passwords** obtained through a data breach of the rockyou service which was used by myspace, and Facebook. Due to the nature of how the wordlist was obtained, the passwords contained within are invaluable as they present a verifiable list of commonly used user passwords.

How to unzip the rockyou.txt wordlist:

By default Kali comes with the **rockyou.txt** wordlist pre-installed, located in the **/usr/share/wordlists** directory as a **rockyou.txt.gz** file. The **rockyou.txt** wordlist needs to be extracted with the **gunzip** tool.

From within the **/usr/share/wordlists** directory where the **rockyou.txt.gz** file is located, one may use the following command to unzip the wordlist:

gunzip rockyou.txt.gz

Note: Once unzipped the **rockyou** wordlist will be extracted as a **rockyou.txt**, one may now use this file to conduct the dictionary attack outlined in the next steps.

aircrack-ng and Wordlist Usage Tutorial:

Once one has managed to capture a **WPA2 four-way handshake**, they may then utilize the **.cap** file they want to attempt to crack using the **aircrack-ng** tool.

The following is a command format example of how aircrack-ng, and a wordlist would be used in conjunction:

aircrack-ng -w [path to wordlist] [capture file] -l [network key file output name] -e [ESSID or network name]

The following is a practical command format example of how aircrack-ng, and the rockyou.txt would be used in conjunction:

aircrack-ng -w /usr/share/wordlists/rockyou.txt CanCom_1686-01.cap -l CanCom_1686key.txt -e CanCom_1686

The following is a breakdown of the previous commands:

aircrack-ng This command specifies we will be using this tool to perform the attack.

-w [path to wordlist] This command option, and the following input field are used to select the wordlist one wishes to use for the dictionary attack against the **.cap** file (in this instance the rockyou wordlist located at the following directory **usr/share/wordlists/rockyou.txt**).

[capture file] This input field is used to specify the **.cap** file of the network four-way handshake that the attack will run against (in this instance it is the **CanCom_1686-01.cap** file).

-l [network key file output name] This command option, and the following input field are used to write the **network key**, aka **network password** to a output file once cracked (in this instance the network password will be written to a file titled **CanCom_1686key.txt**, if sucessfully found).

-e [ESSID or network name] This command option, and the following input field are used to select the **network name** of the network that is associated with the **.cap** file that one is attacking (in this instance it is the **CanCom_1686** network name).

"crunch" Command Options:

Overview: These are some additional options, and techniques that can be leveraged with the **crunch** tool when performing password cracking.

The "-t" Command Option:

-t This command specifies a pattern within a password one is attacking. For example one can specify certain characters that thet want to change in the password string; with the character pattern options they specify for the **-t** command.

"-t" Command Character Pattern Options:

@ Will insert lower case characters.

, Will insert upper case characters.

% Will insert numbers.

^ Will insert symbols.

"-t" Character Option Usage Explaination:

If one were to set the following **@ond%%%** pattern, it would set the following rules. The **@** will insert lower case characters at this location in the string. The following **ond** characters will not change. The

last three **%%%** signs will insert numbers of those positions of the password where they're located.

Character Set Options:

One may also specify character sets for **crunch** to use, if you leave this section blank **crunch** will use the **default character** sets. The maximum, and minimum password lengths **must** be the same size as the password pattern one specifies, or **crunch** will not work.

Character set command format example:

-t @og%%% abcdefg1234567890

The above command means that the **-t** command will replace the **@** symbol with the **lower case** letters one specified in their character set (in the above example **abcdefg**). The **%** symbol will be replaced with the specified numbers in ones character set (in the above example **1234567890**).

Note: The order **must be** lower case characters, uppercase characters, numbers, and lastly symbols. If one does not follow this order they will not get the results they want.

The "-o" Command Option:

-o This command option is used to specify a file to write the output to. This command is **extremely useful** if one is attempting to **brute force** a long password, as it will save ones progress to a file that may be used later to pick up where one left off.

Command format example:

-o [name of the .txt file]

Practical Command format example:

-o networkwordlist.txt

The above command specifise that the output will be written to a file called **networkwordlist.txt**

The "-r" Command Option:

-r This command option tells **crunch** to resume generating words from where it left off. The **-r** command will only work if it's used in tandem with the **-o** command.

One must use the same command as they originally used to generate the words, the only exception to this is when one previously used the **-s** command option.

Note: If one original command used the **-s** option, they **must** remove it prior to resuming the session, to do so one may just add the **-r** command option to the end of the original command.

The "-s" Command Option:

-s This command option is used to specify a starting string, for example: **03god22fs**

Man Pages Note:

The information in this chapter just covers the basics of the **crunch** tool, one should view the **man** page for this tool to get a complete understanding of the command functionality.

Cracking WPS Pins with "reaver" Tutorial:

Overview: The **reaver** tool is used to crack **Wi-Fi Protected Setup (WPS) Pins**, and is extremely effective against **WPA**, and **WPA2** networks.

The following tutorial will explain how **reaver** is used, as well as how one can speed up the pin cracking process.

"wash" and "reaver" Usage Tutorial:

Step One | Finding out ones local wireless network card name using the **ifconfig** command:

First one needs to input the following into their terminal session:

ifconfig

This command will return the status of the currently active network interfaces, using this command one can determine what their wireless interface is (in this case the wireless network interface nane is **wlan0**).

Note: Depending on what OS one is using the wireless network interface name may be different, so one should determine what theirs is and adapt accordingly.

Step Two | Turn off the network card using **ifconfig**:

To proceed with the rest of the tutorial one must turn off their wireless network card.

Command Format Example:

ifconfig [network interface] down

Practical Command Format Example:

ifconfig wlan0 down

The above command will turn off the **wlan0** wireless network interface card, so one may be able to

modify it afterwards.

Step Three | Setting the network card to monitor mode:

Once one has turned off their wireless network card, they may now set it to **monitor mode** by typing **iwconfig** followed by the wireless network interface name, which would then be followed by the **mode** command, and lastly the **monitor** command option.

Command Format Example:

iwconfig [network interface] mode monitor

Practical Command Format Example:

iwconfig wlan0 mode monitor

The above command would set the **wlan0** network interface to **monitor mode**.

Step Four | Turn the network card back on using **ifconfig**:

Once one has set their wireless interface card to **monitor mode**, they can then turn it back on by typing **ifconfig** into the terminal, followed by the wireless network interface name, and lastly the **up** command option.

Command Format Example:

ifconfig [network interface] up

Practical Command Format Example:

ifconfig wlan0 up

The above command will turn on the **wlan0** interface.

Step Five | Using **airmon-ng** to kill processes:

Once ones network interface card is back up in **monitor mode**, one may then type the following command into the terminal **airmon-ng check**, followed by their wireless network interface name.

Command Format Example:

airmon-ng check [network interface]

Practical Command Format Example:

airmon-ng check wlan0

Using the previous command will now bring up a list of processes associated with the **wlan0** network interface card that could cause problems.

The next step is to kill these processes, this should be done in a specific order, otherwise the processes may be able to respawn one another.

To kill a process utilizing **airmon-ng**, one needs to type in **kill** followed by the **PID number** of the process they would like to kill. For instance **kill 1048** would kill the process associated with the 1048 number.

Command Format Example:

kill [PID #]

Practical Command Format Example:

kill 1048

Note: Once a process has been killed one needs to restart the **airmon-ng check** process to kill the rest.

Order to kill processes:

1). NetworkManager
2). dhclient

The rest of the processes can be killed in any order.

Step Six | Using **wash** to find vulnerable WPS networks:

The **wash** tool is used to perform a specific network scan to determine whether a wireless access point is vulnerable to a **reaver** WPS pin bruteforce attack.

To perform a "wash" scan one may use the following command format:

wash -i [your network interface]

This is an explanation of the previous command:

wash This command specifies that this tool will be used to perform the WPS network scan.

-i This command option is used to specify an interface that will be used to capture packets.

[your network interface] This input field is where one would specify the network interface they will be using (for instance **wlan0**).

Once the scan has been performed successfully one will be presented with several different fields of information, the most important of which being the **WPS Locked** field!

Note: For an explanation of the relevent WPS scan fields one can refer to the **WPS Scan Field Names** section, in the **Attacking Wireless Networks Overview** chapter.

Step Seven | Using **reaver** to perform a bruteforce WPS pin attack:

Once one has used the **wash** command to find a network that is susceptible to a **reaver** Wi-Fi Protected Setup pin bruteforce attack.

The following is a command format example of how to attack a network where "WPS" is enabled:

reaver -i [your network interface] -b [BSSID] -r [recurring delay 0:000] -c [channel number] -vv

The following is a practical command format example of how to attack a network where "WPS" is enabled:

reaver -i wlan0 -b A8:39:44:47:D4:AC -r 2:120 -c 6 -vv

The following is an explanation of the previous command:

reaver This tool will be used to perform the WPS pin bruteforce attack.

-i [your network interface] This command option, and the following input field is used to specify the network interface one will be using for the attack (for example **wlan0**).

-b [BSSID] This command option, and the following input field is used to specify the mac address, or BSSID of the target network one will be attacking.

-r [recurring delay 0:000] This command option, and the following input field is used to specify that a recurring delay is to be set for the WPS pin attack.

The following is an example of the accepted format to set this delay **2:120.** In this instance the **2** number would be the pin attempts performed, while the **120** number is the seconds specified to "sleep", or wait, before the next set of pin attempts take place.

Note: This recurring delay is important to set to prevent the AP from locking one out due to too many attempts at cracking the pin too quickly!

-c [channel number] This command option, and the following field is used to specify the channel number of the target network one wishes to attack.

-vv This command option is used to specify the verbosity output level, and is useful to include to understand potential issues when debugging (in this instance **-vv** would provide one with double

verbosity output, but one may also specify more, or less of a verbosity level by adding, or removing the **v character**).

Important "reaver" Notes:

WPS pin bruteforce attacks conducted with **reaver** are no longer as effective as they once were, this is due to newer routers being able to better identify, and protect against these attacks as they occur.

However **reaver** is still a useful tool when used in conjunction with **Pixiewps** in order to attempt a **pixiedust** attack, more information on "pixiedust" attacks will be covered in the upcoming chapter titled "**Using Reaver, or Bully to Perform a WPS Pixiedust Attack Tutorial**".

Using "reaver" or "bully" to Perform a WPS Pixiedust Attack Tutorial:

Overview: Unlike a standard bruteforce attack against all **11000 potential WPS** pins using **reaver**; a pixiedust attack uses a different vulnerability against WPS.

A pixiedust attack works by bruteforcing a badly generated key, because of a flaw in how the random numbers are generated in many routers, below is a complete explaination of how a pixiedust attack works:
A WPS pin consists of 8 digits - two pre-shared keys, or PSK's, each PSK has half the pin.

To understand how a Pixiedust attack works, one needs to understand how the requests to the A.P. work:

1). Computer sends - EAPOL Start.

2). Router sends -EAP-Request for the identity.

3). Computer sends -Responds with the identity.

4). Router sends -EAP request.

5). Computer sends -EAP response.

The process loops these requests a few more times before the credentials are sent.

However, during this process, the operator computer has been given the following:

- Diffie Hellman Public Key of the Enrolee.
- Diffie Hellman Public Key of the Registrar.
- Two hashes - of the **WPS PIN**.
- Enrolee nonce and a derived authkey.

In order to successfully bruteforce the previously mentioned PSK's, one will need two more nonces,

which are supposed to be randomly generated.

This is the most important part, since the random numbers are not really random, but are derivations of the hashes (or are just zeros), then one can proceed to bruteforce this key, even on a slow system.

This attack will work if the implementation on the router is flawed (which it is in most cases pertaining to legacy routers), and one should be able to find a list of routers vulnerable to this attack online.

With the theory of how this attack works now discussed, the following sections discuss how to perform the attack using the **reaver**, and **bully** tools.

"reaver" Pixiedust Attack Command Format Example:

reaver -i [your network interface] -b [BSSID] -c [channel number] -r [recurring delay 0:000] -S -K 1 -vv

"reaver" Pixiedust Attack Practical Command Format Example:

reaver -i wlan0 -b 18:9C:27:B9:78:BE -c 1 -r 2:120 -S -K 1 -vv

This is an explanation of the above command:

reaver This is used to specify this tool is to be used perform the pixiedust WPS attack.

-i [your network interface] This command option, and the following input field is used to specify the network interface one will be using for the attack (for example **wlan0**).

-b [BSSID] This command option, and the following input field is used to specify the mac address, or BSSID of the target network one will be attacking.

-c [channel number] This command option, and the following field is used to specify the channel number of the target network one wishes to attack.

-r [recurring delay 0:000] This command option, and the following input field is used to specify that a recurring delay is to be set for the WPS pin attack.

The following is an example of the accepted format to set this delay **2:120.** In this instance the **2** number would be the pin attempts performed, while the **120** number is the seconds specified to "sleep", or wait, before the next set of pin attempts take place.

Note: This recurring delay is important to set to prevent the AP from locking one out due to too many attempts at cracking the pin too quickly!

-S This command specifies **reaver** to use **small DH keys** to improve crack speed.

-K 1 This command specifies **reaver** to run a **pixiedust** attack with **PKE**, **PKR**, **E-HASH1**, **E-HASH2**, and

E-Nonce (Ralink, Broadcom, Realtek). Increment the value after **-K**.

-vv This command option is used to specify the verbosity output level, and is useful to include to understand potential issues when debugging (in this instance -vv would provide one with double verbosity output, but one may also specify more, or less of a verbosity level by adding, or removing the **v character**).

Important Note: In order to perform a **pixiedust** attack, one **must include** the **-K** command option, otherwise a pixiedust attack will **not** be attempted.

"bully" tool overview: The **bully** tool is similar to **reaver** in that it was designed to attack WPS, both using brute force, and pixiedust attack methods.

It has several advantages over the original **reaver** code. These include fewer dependences, improved memory, and cpu performance, correct handling of endianness, and a more robust set of options.

As such, **bully** stands as a similar, but fantastic alternative to the **reaver** tool.

"bully" Pixiedust Attack Command Format Example:

bully -b [BSSID] -c [channel number] -B -d -v [verbosity level] [network interface]

"bully" Pixiedust Attack Practical Command Format Example:

bully -b 18:9C:27:B9:78:BE -c 1 -B -d -v 3 wlan0

This is an explanation of the above commands:

bully This is used to specify this tool is to be used perform the pixiedust WPS attack.

-b [BSSID] This command option, and the following input field is used to specify the mac address, or BSSID of the target network one will be attacking.

-c [channel number] This command option, and the following field is used to specify the channel number of the target network one wishes to attack.

-B This command option is used to specify the **bully** tool to attempt to bruteforce the WPS pin checksum digit.

-d This command option is used to specify the **bully** tool to attempt a **pixiedust** (or **pixiewps**) attack.

-v [verbosity level] This command option, and subsequent input field is used to specify the verbose output that will be displayed. The verbosity level can be between **1-4**, with **1** being the quietest, and **4** being the most in depth.

[network interface] This input field is used to specify the local network interface one previously set to

monitor mode (for instance **wlan0**).

Accessing an A.P. Upon Successful WPS Attack with “reaver”, or “bully”:

Once one has successfully been able to acquire a WPS pin using a **reaver**, or **bully** attack, they will now be able to use this pin to reveal the AP password.

Do to so one can input the WPS pin to reveal the AP password using the following command formats in either **reaver**, or **bully.**

“reaver” WPS Input Command Format Example:

reaver -i [network interface] -b [target BSSID] -p [target WPS pin] -vvv

“reaver” WPS Input Practical Command Format Example:

reaver -i wlan0 -b 10:7B:EF:81:4B:AE -p 48293012 -vvv

The above command demonstrates how one would input a WPS pin using the **reaver** tool.

To input a WPS pin one would use the **-p** command option, followed by the WPS pin they would like to input.

Note: If the pin one inputs is correct, it will reveal the AP password upon executing the command.

“bully” WPS Input Practical Command Format Example:

bully -b [target BSSID] -c [target channel #] -p [target WPS pin] -v [verbosity level] [network interface]

“bully” WPS Input Practical Command Format Example:

bully -b 18:9C:27:B9:78:BE -c 1 -p 48293012 -v 3 wlan0

The above command demonstrates how one would input a WPS pin using the **bully** tool.

The use of the **-p** command is the exact same with the **bully** tool as it is with **reaver**. The only difference is the **bully** command format is different.

Note: if the pin one inputs is correct, it will reveal the AP password upon executing the command.

Exploiting Router Vulnerabilities via “exploit-db.com”:

Overview: Routers are the first line of defense for a network. Compromising a router, will enable the attacker to leverage almost any type of attack on the network.

Examples of attacks that may be conducted against a network where the router has been hacked:

- Excluding access to others from the firewall rules, and accessing said rules directly.

- Change the DNS settings to redirect traffic, and serve the information the attacker wants to present to the victim.

- Setting up proxues on the compromised network.

Open Source Intelligence Gathering, and "exploit-db.com" Usage:

There are several methods to use **Open Source Intelligence (OSINT)**, to enumerate information on a router, to see if there are any vulnerabilities to exploit. Once one has obtained information against the target router, they may then attempt to leverage this information to search for relevant exploits via the **exploit databse** website, which is a website that hosts a collection of exploits.

In the next sections of this chapter various router enumeration methods will be discussed, before an overview of how to leverage the exploit database website resource is given.

"Banner grabbing" router model identification method:

One method one can use to identify a router for later use of that aquired information to investigate for exploits is "banner grabbing".

This term refers to the use of pulling open source information from the router directly, through the router landing page. Most of the time this can be easily achieved by simply examining the information provided directly by the said page, or alternatively by viewing the said page "**source code**".

Note: The most useful information to find to leverage for router exploitation is the specific router model number, and the maker.

"nmap" router identification method:

The **nmap** tool can be used to identify, as well as exploit a router. This tool, and method will be discussed in the next chapter "**Exploiting Router Vulnerabilities via nmap".**

Searching for exploits via "google.com":

Once one has found their target router model information; they may then use the **google** search engine to explore for said router model exploits.

To do so, one may input the router model information, followed by the word "**vulns"**, or "**exploit**" into the google search engine, then press **enter**.

Searching for exploits via www.exploit-db.com:

The **exploit database** domain is a website dedicated to archiving published exploits, and can be accessed by visiting the following link:

https://www.exploit-db.com

To use the site, one simply needs to enter the specific name of the device they wish to exploit into the website search bar.

If there is a potentially relevant exploit published on the site, it will be listed for the operator after conducting the search. The returned search output will provide one with more information on any listed exploits, and how they're executed, once one selects the specific exploit they're interested in using (via the listed exploit links).

Exploting Router Vulnerabilities via "nmap":

Overview: The **nmap** tool is an open source network scanner, created by **Gordan Lyon (aka Fyodor)**.

nmap provides a number of features for probing computer networks, including **host discovery**, **service**, and **operating system** detection.

These features are **extensible** by scripts that provide more advanced service detection, vulnerability detection, and other features.

Note: For the purposes of this tutorial, one will conduct a network scan in order to try and determine their router model information.

Afterwards, one will try to execute some **Nmap Scripting Engine (NSE)** scripts against the router to attempt to find, and exploit vulnerabilities.

While exploitation is not always possible, **nmap** is still a fantastic tool for reconnaissance, system hardening, and pentesting puposes.

Step One | Using the **route** tool to find a gateway IP:

In order to find the gateway IP of ones router, one may use the following command:

route -n

Once executed the router gateway IP will appear underneath the "**Gateway**" field that corresponds to the **0.0.0.0** destination.

Note: Some newer distributions of linix may require another method to obtain the gateway IP, if this

command does not work one should conduct further research to determine which command will function properly for this purpose.

Step One B (Optional) | Using the **nslookup** tool to find a network IP:

The **nslookup** tool can be used to determine the IP address of a domain name server, or vice versa.

Note: For the purposes of this tutorial, one does not need to perform this step (as one will be attacking their own local gateway in the next step), however this step is included to provide a reference of the nmap attack process against an external target in practice.

Command Format Example:

nslookup [DNS or IP address]

Practical Command Format Example:

nslookup scanme.nmap.org/

The following is an explaination of the above command:

nslookup This is the command used to specifies that this tool will be used to conduct the lookup.

[DNS or IP address] This input field is where one would input the **DNS** name, or **IP address** they wish to query.

Once the command has been executed, one should be able to get the IP, or domain of their specified query. Normally one should take note of this address as it would be needed for the next step against an external target.

Step Two | Using **nmap** against an IP for reconnaissance:

The **nmap** tool is useful for **network reconnaissance** in order to determine what devices are on the network; and what ports are open. Then information can then be leveraged to determine if there are any exploits, or vulnerabilities that can be executed against the network; or associated network devices.

Note: For the purposes of this tutorial one will be running a **nmap** scan their local router gateway IP, however a network scan can be can be conducted against **ANY** IP.

Command Format Example:

nmap -sS -sV -O [target IP address]

Practical Command Format Example:

nmap -sS -SV - O 192.168.0.1

The following is an explanation of the above command:

nmap This command is used to specify this tool will be used to conduct the network scan.

-sS This command option specifies that a **system scan** of the network will be conducted.

-sV This command option specifies that a **system version scan** will be conducted, this kind of scan provides version information for devices/services on the network.

-O This command option specifies that a **operating system scan** will be conducted, this provides one with the OS information for the devices on the network.

[IP address here] This input field is where one would input the target IP address they want to run a network scan against.

Note: In the practical command format example the **192.168.0.1** address was used as an example, as this private IP address is commonly used as the default login address (gateway) for routers, and modems from brands like TP-Link, D-Link, Netgear, and WAVLINK. To determine ones local IP address one may use the following command **route -n**

Once one has executed the scan against the target IP, they should take note of the scan output returned (especially in this instance, the specific router model information).

This router information will be important for vulnerability assessment, and the exploitation one will be attempting in the following step.

Step Three | Vulnerability assessment, and exploitation utilizing **nmap (NSE) scripts**:

Once one has enough information on the target network, and its associated devices, one can now check the **Nmap Scripting Engine** documentation to explore for scripts that can be executed against the target network devices, and services, to investigate potential vulnerabilities, and utilize exploits.

To review the NSE scripts, and obtain a description of each kind of script (the side bar conveniently lists the NSE scripts by category) one can use the following resource link:

https://nmap.org/nsedoc/

Once one has selected a NSE script they would like to run against the target network, one should click the script link for more information on the script, and an explanation on how to execute the script; and the acceptable arguments for it.

For the purposes of this tutorial, it will be assumed that the network scan conducted detected a IP router in use. Using this information on the router, one will now begin searching through the NSE scripts to see if there are scripts one can use against a specific **T.P link** router.

For example one can run the following script against a TP link router to determine whether it is vulnerable to exploitation:

nmap -p80 --script http-tplink-dir-traversal.nse --script-args rfile=/tmp/ath0.ap_bss -d -n -Pn [target IP]

Practical Command Format Example:

nmap -p80 --script http-tplink-dir-traversal.nse --script-args rfile=/tmp/ath0.ap_bss -d -n -Pn 192.168.0.1

The following is an explanation of the above command:

nmap This command is used to specify this tool is to be used for this attack.

-p80 This command option specifies that one will be scanning port 80 (this port is often used by the HTTP service).

--script This command option specifies that one will be using an NSE script.

http-tplink-dir-traversal.nse This is the specific script that was specified for this attack.

--script-args This command specifies that script arguments will be executed in the command.

rfile=/tmp/ath0.ap_bss This is the script argument that has been selected to run.

-d This command specifies that the tool debigging level will increase.

-n This command specifies **nmap** to never do a reverse DNS query.

-Pn This command specifies that **nmap** will skip the discovery stage, and treat all ports as open).

[target IP] This is where one would input the IP of the target they wish to attempt to exploit, in this instance the **192.168.0.1** **gateway IP**.

Once the attack has been executed, one will be able to determine if the router is susceptible to a **directory traversal** attack. This kind of attack can lead to sensitive information disclosure, such as passwords, system configuration files, ect.

For more information on this attack one may view the associated NSE script documentation:

https://nmap.org/nsedoc/scripts/http-tplink-dir-traversal.html

Best Router Security Practices and Suggestions:

Overview: The following is a list if the best practices, and suggestions to secure a router against hacking attacks.

1). Change the default router password:

If ones router has a default password, it should be changed immediately, as well as the user name. The password should be at least 10 characters long, and include letters, numbers, symbols, lower, and uppercase characters.

2). WPS pin authentication should be disabled:

Having WPS pin authentication enabled makes it much easier for an attacker to hack a vulnerable router, for this reason WPS should be **disabled** via the router settings.

3). Use WPA2 instead of WEP encryption:

WEP encryption is extremely vulnerable, and should not be used under any circumstances, WPA2 (or ideally WPA3) shoild be used instead.

4). Replace vulnerable routers immediately:

If a router is determined to have an applicable exploit found via the **exploit-db** website, or an **nmap** NSE script, the router should be replaced immediately. As an exploitable router can compromise ones entire network security.

5). Diable the router when not in use:

If one disables the router when it is not in use, they make it substantially more difficult for the attacker to compromise the network, through a **clientless WEP**, or **PMKID WPA2** based attack.

Best Network Security Practices and Suggestions:

Overview: Security practices for a network differ from router practices, in that they require more awareness from the user about the network itself.

1). Use a strong WPA2 password:

Choose a strong Wi-Fi password that can not be easily guessed. The password should be at least 12 characters long and include letters, numbers, symbols, lower, and uppercase characters.

2). Hide the network name:

Wi-Fi access points are usually configured by default to broadcast the name of the local wireless network (known as the service set identifer, or SSID), to make it easy to find and connect to. However, the **SSID** can also be set to **hidden** so that one needs to know the name of the network before they are able to connect to it.

Given that employees should know the name of the company Wi-Fi network (abd the same goes for friends, and family in a household), it makes no sense to broadcast it so that anyone else who happens to be in the area can easily find it too.

Note: It is important to note that hiding your SSID should never be the only measure one takes to secure their Wi-Fi network, because hackers using Wi-Fi scanning tools such as **airodump-ng** can still detect the network, and it's **SSID** even when hidden.
However security is all about providing multiple layers of protection, and by hiding ones SSID one may prevent attracting the attention of opportunistic hackers, as such it is a simple security measure worth taking.

3). Provide a seperate network for guests:

If one wants to allow visitors to use their Wi-Fi, it's sensible to offer a **guest network**.

This means that the visitor can connect to the internet without getting access to the company, or households internal network. This is important for both security reasons, and also to prevent the visitor inadvertently infecting ones network with viruses, or other malware.

One way to do this is to use a seperate internet connection with its own wireless access point. In fact this process is rarely necessary as most business grade (and a lot of newer consumer) wireless routers have the capability of running two Wi-Fi networks at once - The main network, and another network specifically for guests (often with the SSID Guest)

It still makes sense to turn on WPA protection on the guest network, rather than to leave it open, for two important reasons:

The first is to provide some level of control over who uses it, one can provide the password to guests on request, and as long as one changes the password frequently, they can prevent the amount of people who know the password from growing too large.

More importantly, this protects ones guests from other people on the guest network who nay try to snoop on their traffic using a tool such as **wireshark**. This is because even though they are ysing the same WPA password to access the network, each users data is encrypted with a different **session key,** which keeps it safe from other guests.

4). Enable MAC authentication network users:

One can limit who accesses their wireless network even further by only permitting certain devices to connect to it, and barring the rest. Each wireless device will have a unique serial number known as a

MAC address, and MAC authentication only allows access to the network from a set of addresses defined by the administrator.

This prevents unauthorized devices from accessing network resources, and acts as an additional obstacle for hackers who might want to penetrate ones network.

5). Use a firewall:

Hardware firewalls provide the first line of defense against attacks coming from outside of the network, and most routers have firewalls built into them, which checks data coming into, and going out, and blocks any suspicious activity. These devices are usually set with reasonable defaults that ensure they do a decent job.

Most firewalls use packet filtering, which looks at the header of a packet to figure out its source, and destination addresses. This information is compared to a set of predefined and/or user-created rules that govern whether the packet is legitimate, or not, and thus whether it's to be allowed, or discarded.

Software firewalls usually run on the endpoint desktop, or laptop, with the advantage of providing a better understanding of what network traffic is passing through the device.

More than just which ports are being used, and can allow, or block that programs ability to send, and recieve data. If the software firewall isn't sure about a particular program it can ask the user what it should do before it blocks, or allows traffic.

6). Rogue Wi-Fi access point awareness:

Rogue access points present a huge security risk. These aren't ones official company Wi-Fi access point, rather these are an access point brought in by employees (perhaps because they can't obtain a good Wi-Fi signal in their office), or conceivably by hackers who have entered the premise, abd surreptitiously connected a rogue access point to a ethernet port, and hidden it.

In either case, rogue access points present a threat, because one has mo control over them, or how they are configured. For example, one could be set up to broadcast the network SSID (the 32 character identifier for a wireless network), and allow anyone to connect without providing a password.

To detect rogue access points one needs to scan their offices, and the area around it on a regular basis using a laptop, or mobile device, equipped with suitable software such as **Vistumbler** (a wireless network scanner), or **airodump-ng**.

These programs allow the laptop to "**sniff"** the airwaves, to detect any wireless traffic traveling to, or from a rogue access point, and help one identify where they are located.

For more information on the **Vistumbler** tool please refer to the following resource:

https://www.vistumbler.net/

7). Use a VPN:

A VPN, or Virtual Private Network will help one stay safe, and secure online by ensuring you private communications remain private.

VPNs can keep ones data hidden from prying eyes from one end to the other by encrypting it.

In theory, hackers could penetrate a network, and they would still not be able to intercrpt ones traffic assuming that a VPN is running permanently.

Conclusion:

Once again congratualtions on making it to the end, and thank you for reading!

The final course in this serious will cover advanced wireless attacks against newer Wi-Fi protocols, the **airgeddon** Wi-Fi hacking framework, captive portal social engineering attacks, alternative password hacking tools, and newer attack vectors such as the **PMKID WPA2** attack, as well as pin database attacks, and more!

Additionally, there is very little content that discusses post exploitation once a wireless network has been compromised, so the final course will also provide an overview of some of the more common Wi-Fi post exploitation methods one can use for these purposes.